TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Research Group

8 Time Series Data

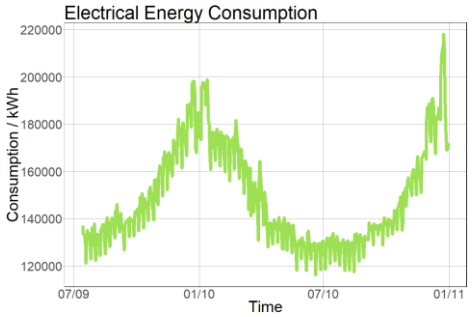
Scalable Data Engineering

Version: Thursday, December 1, 2022

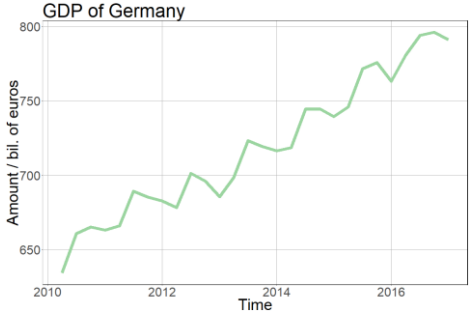
Time Series Occur in Many Domains

Dresden Database Research Group

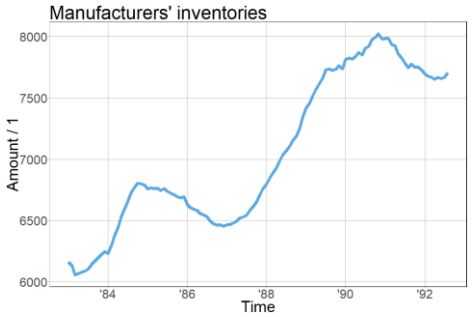
Utilities



Economy



Industry



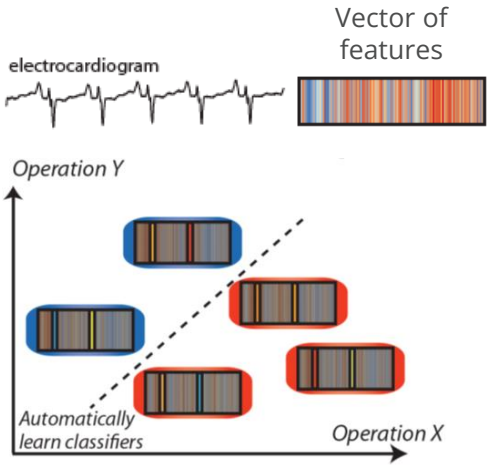
Agriculture Animal Population Yield	**Chemistry** Chemical Concentration	**Computing** Online Advertisement Query Processing	**Crime** Robberies Drunkenness	**Demography** Population Rates Immigrants
Economic GDP Inflation	**Finance** Stock Development Price Development	**Health** Infections Suicide Rate	**Industry** Production Planning Inventory Planning	**Meteorology** Temperature Rainfall
Physics Earthquakes Sunspots	**Politics** Election Outcomes Sports	**Sports** Player Performance Team Performance	**Transport** Tourist Visits Airline Passengers	**Utilities** Energy Balancing Gas Production

Goals in Time Series Analytics

Dresden Database Research Group

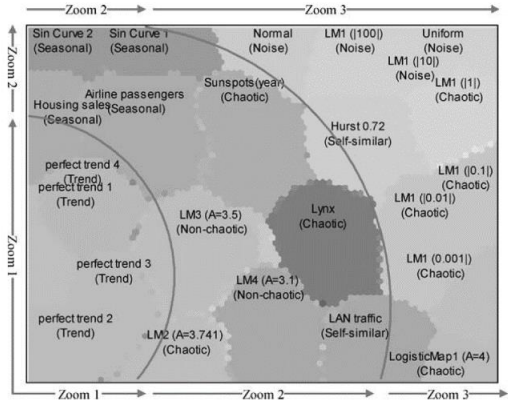
Time Series Classification

- Detect characteristics such as, e.g., features that describe series classes



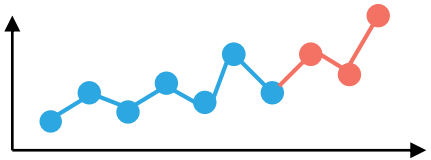
Time Series Clustering

- Group time series with respect to distance criteria



Time Series Forecasting

- Fit a model to a given time series and other influences
- Extrapolate series for prediction



Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods.
Wang, X., Smith, K., & Hyndman, R. J. (2006). Characteristic-based clustering for time series data. Data Mining and Knowledge Discovery.

Agenda

Dresden Database Research Group

Basics of Time Series

- Time Series Models
- Decomposition
- Missing Data & Imputation
- Time Series Engineering

Applications

- Time Series Classification
- Time Series Clustering
- Time Series Forecasting

Time Series Storage

- Relational Storage
- Array Storage

Time Series Models

Time Series Model

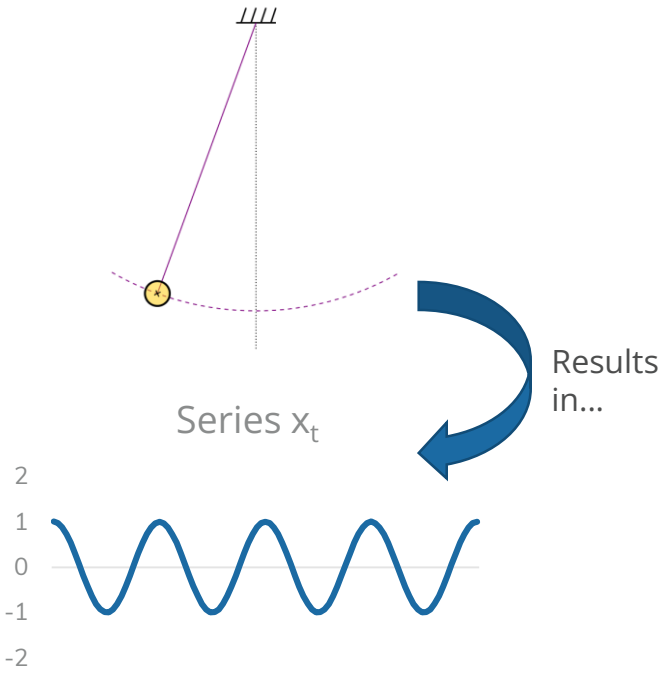
Idea

- Consider a time series as an unknown process combined with an unpredictable deviation
- It is measured at equidistant time instances and results in a sequence
- The unknown process is represented by a descriptive model P_t
- The unpredictable deviation res_t is assumed to be random, with
  - Expected value is 0
  - Variance is constant
  - Normally distributed
- Thus, the time series is represented by a descriptive model:

$$x_t = P_t + res_t \qquad (P_t\text{...process}, res_t\text{...residuals})$$

$$\underline{x}^T = (x_1, \dots, x_t, \dots, x_T)$$

Example: Pendulum



Terminology

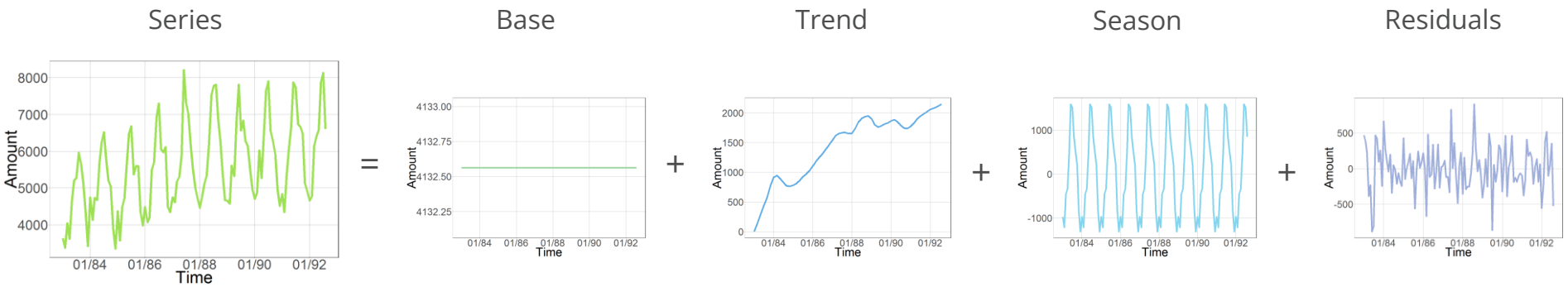
Dresden Database Research Group

Time series components

- Base: stationary part of the time series
- Trend: long-term change in the mean level
- Season: cyclical repeated behavior
- Residuals: unstructured information assumed to be random

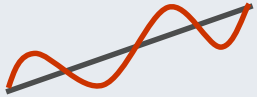
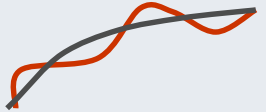
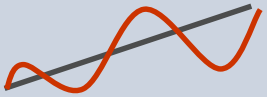
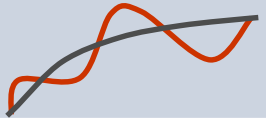
Often, base is part of the trend component!

Additive composition $x_t = base_t + trend_t + season_t + res_t$



Composition Types

Dresden Database Research Group

	No Trend	Linear Trend	Non-linear Trend
No Season			
Additive Season			
Multiplicative Season			

The most common models

Pegels, C. C. (1969). Exponential Forecasting: Some New Variations. *Manage. Sci.*, *15*(5), 311–320.

Decomposition

Classical Decomposition

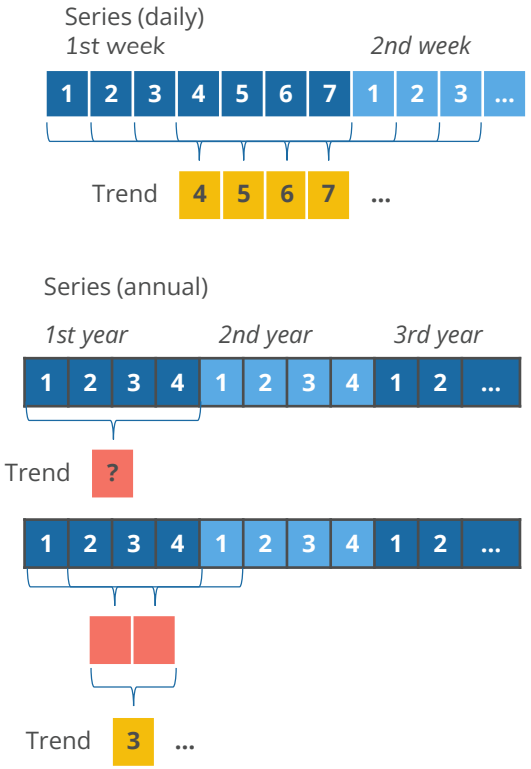
Dresden Database Research Group

Basic idea

- Moving-average filter of continuous windows of size N (N is odd)
- $tr_t = \frac{x_{t-(N-1)/2} + \ldots + x_t + \ldots + x_{t+(N-1)/2}}{N}$
- Extraction of trend by windows that take into account the season length
- Extraction of season by averaging each time instance of the same seasonal position (all Mondays, all Tuesdays,...)
- Disadvantage: does not decompose the endpoints

Average centering

- A technique needed if season length is even
- Take two moving averages and average their result



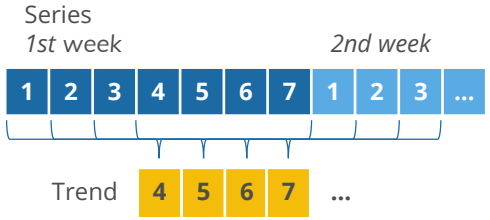
A season of length 7 such as
- Monday (1)
- Tuesday (2)
- ...

Classical Decomposition (2)

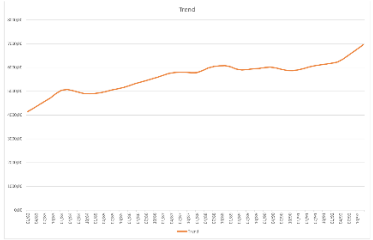
Dresden Database Research Group

Retrieval of trend

- Moving-average filtering
- In case of even season length, centralize first



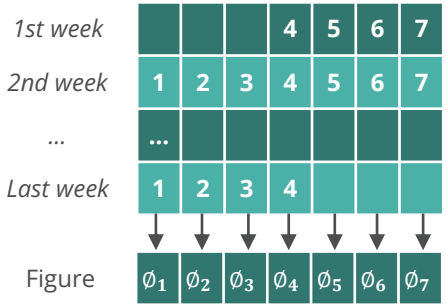

```
trend <- filter(series, rep_len(1,7)/7)
```



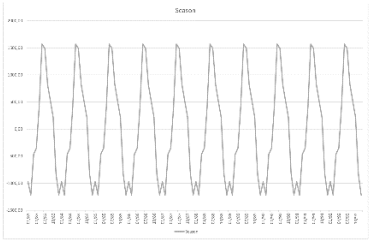
Season retrieval

- Detrend series
- Figure represents the average of each time instance of a season
- De-mean figure

```
detrend <- series - trend
```



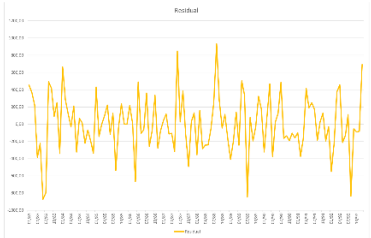

```
season <- figure - mean(figure)
```



Residuals

- Subtract components

```
residuals <- series - season - trend
```



Further Decomposition Techniques

Dresden Database Research Group

X11

- Method published by the U.S. Bureau of the Census and used adopted by several statistical agencies
- Based on classical decomposition with several moving-average steps
- Advantages
  - Endpoints decomposed using predictions from ARIMA forecasting method
  - Extensions for holiday effects
  - Support slowly varying season representing changes in seasonal behavior
- Disadvantage: Only for quarterly and monthly time series

STL

- Application of regression technique (*Loess smoothing*)
- Recursive application of smoothing and robustness checks
- Advantages
  - Support arbitrary season lengths
  - Versatile and robust decomposition technique
- Disadvantage: Parameters are not automatically retrieved

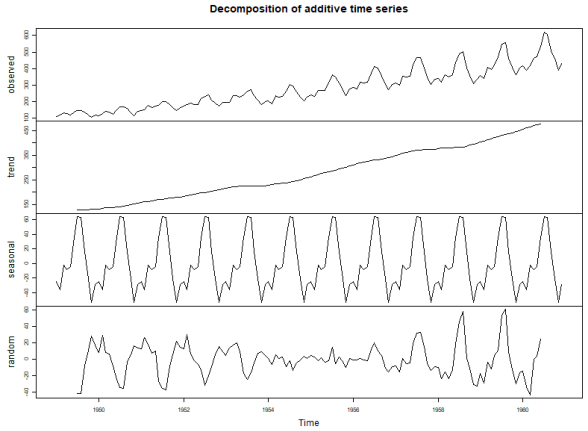
Decomp. Features	DEC	X-11	STL
Arbitrary season length	✓	-	✓
Slowly varying season	-	✓	✓
Robustness	-	✓	✓
Endpoints	-	✓	✓

Examples in R

Dresden Database Research Group

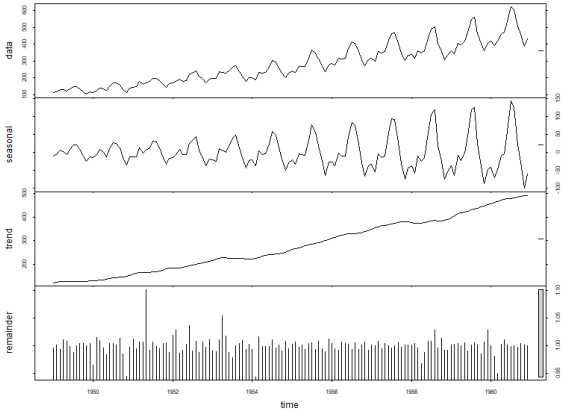
Classical decomposition

```
decomposed <-
decompose(AirPassengers)
```



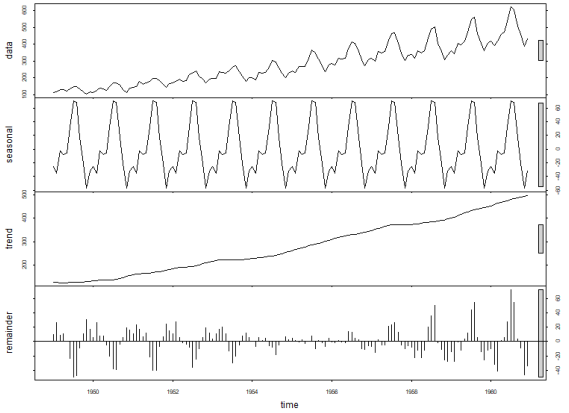
X11

```
library(seasonal)
decomposed <-
seas(AirPassengers, x11 =
"")
```



STL

```
decomposed <-
stl(AirPassengers,
s.window = "period")
```



Missing Data & Imputation

Missing Data

Dresden Database Research Group

Missing data

- All kind of incomplete observations
- Item imputation: Missing answers in surveys
- Unit imputation: Missing data points, e.g. in time series

Caused by

- Incompletely answered surveys (willingly and accidently)
- Technical problems, e.g., sensor failure, transmission errors, data loss

Missing data treatment

- Complete case studies, delete all cases with missing values
- Imputation techniques, fill missing observations

Complete Case Studies

Dresden Database Research Group

Procedure

- Delete all cases with missing observations
- Available Case Studies, only neglect cases with missing values for the current analysis task
  - Varying sample sizes in different analysis tasks!

Advantage

- Easy to implement

Disadvantage

- Missing values have to be randomly distributed over the data set, otherwise deletion introduces bias
- Decrease reliability of analysis by decreasing sample size

Not feasible for time series

- Aggregates would be too low
- Series where forecasts are requested are missing

Imputation

Dresden Database Research Group

Goal

- Replace missing values with substitute values

Base data

- Hot Deck – Use current data set for the replacement value calculation (most common)
- Cold Deck – Use another data set (e.g. older survey answers) for replacement value calculation

Singular imputation

- Use only one technique to calculate replacement values

Multiple imputation (Ensemble)

- Apply several single imputation techniques to impute the same value
- Combine their replacement values in the final result

Imputation for Time Series

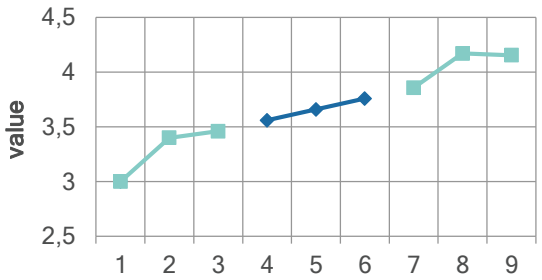
Dresden Database Research Group

Substitute values

- Pick value from the same time series at an earlier time
  - e.g. 9.00am values from Tuesday when 9.00am from Wednesday is missing
- Pick value from other similar time series (similar behavior, similar attributes, similar value range)

Linear interpolation, mean interpolation

- Fill small gaps via linear interpolation or mean of known values
- Not suited for large gaps since fluctuations are eliminated



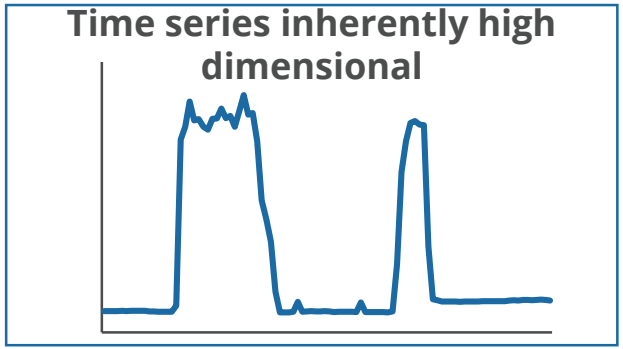
Aggregate extrapolation

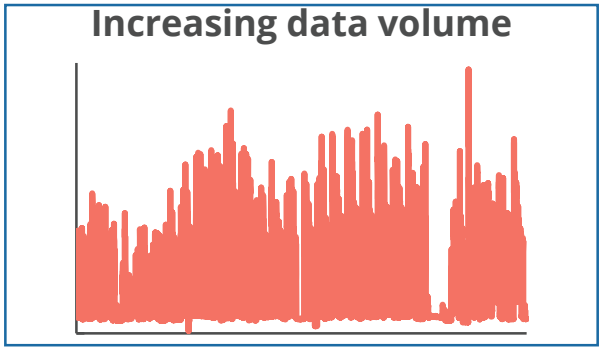
- Extrapolate Aggregate of time series based on portion of monitored series
- Horvitz-Thompson Estimator uses probability of a time series to be missing π and the expectation value μ

$$agg = \sum_i {\pi_i}^{-1} \cdot \mu_i$$

Time Series Engineering

Curse of Dimensionality





- How to capture the most meaningful information?
- How to prepare this information for data-mining tasks?

Time Series Engineering

Raw-value-based | **Shape-based** | **Feature-based** | **Model-based**

Shape-based Representation

Dresden Database Research Group

Representation

- Transform a series to segments
- Time-dependent representation
- High compression

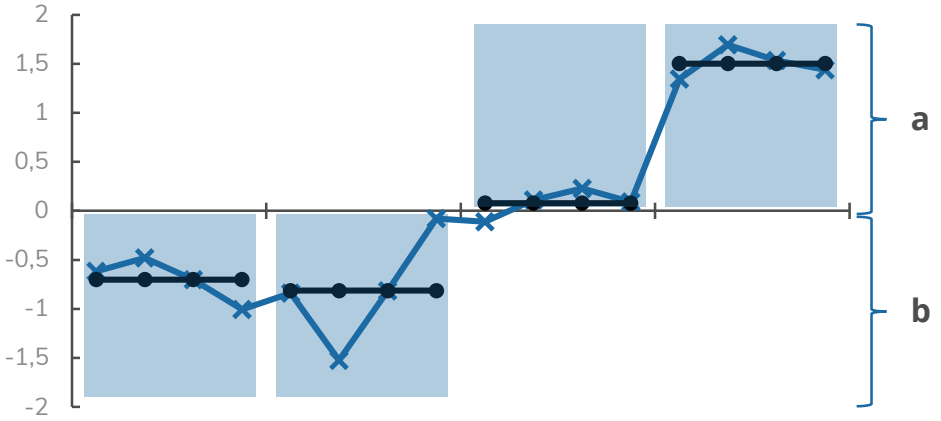
Piecewise aggregate approximation (PAA)

- Represent each segment by its mean
- $\underline{\bar{x}}^T = (\bar{x}_1, \ldots, \bar{x}_w, \ldots, \bar{x}_W)$ (W segments per series)

Symbolic aggregate approximation (SAX)

- Discretize the mean into an alphabet
- $\underline{\hat{x}}^T = (\hat{x}_1, \ldots, \hat{x}_w, \ldots, \hat{x}_W)$
- $\hat{x} \, \epsilon \, \{1, 2, \ \ldots, A\}$ (A … size of alphabet)
- Symbols of alphabet representation by alphanumeric characters („a“, „b“, …)
- Assume normally distributed PAA and split the real numbers into segments of equal size

Segments



PAA:	-0.70	-0.81	0.08	1.56
SAX:	b	b	a	a

(segments per series = 4, alphabet size = 2, breakpoint at 0.0)

Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Al- gorithms. In Workshop Proc. of SIGMOD, 2003

Shape-based Distance

Dresden Database Research Group

PAA distance measure

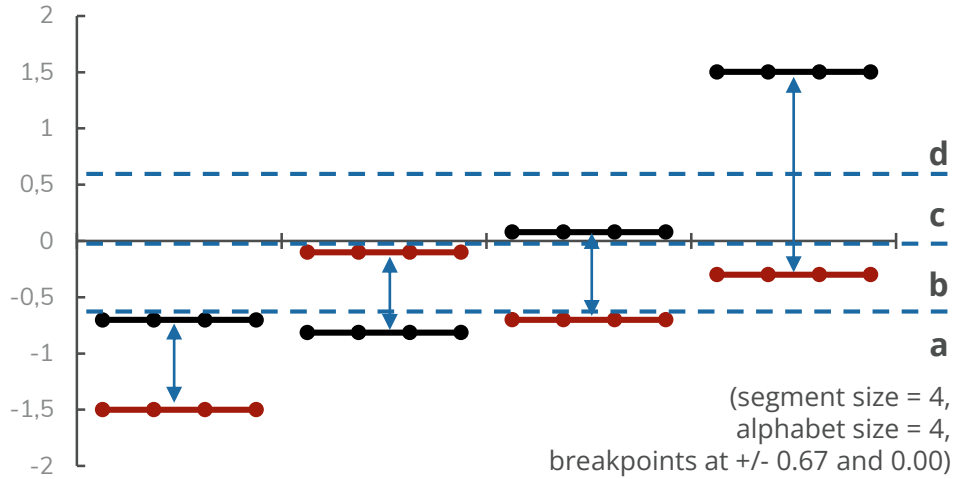
- $d_{PAA}(\bar{\underline{x}}^i, \bar{\underline{x}}^j) = \sqrt{T/W} \cdot \sqrt{\sum_{w=1}^{W} (\bar{x}_w^i, \bar{x}_w^j)^2}$

SAX distance measure

- $d_{SAX}(\hat{\underline{x}}^i, \hat{\underline{x}}^j) = \sqrt{T/W} \cdot \sqrt{\sum_{w=1}^{W} cell(\hat{x}_w^i, \hat{x}_w^j)^2}$
- *cell* returns the minimum distance of two symbols:

	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

Segments



SAX

$\hat{x}^1$	a	a	c	d
$\hat{x}^2$	a	b	a	b

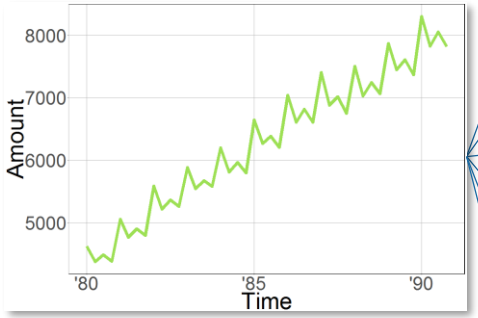
Feature-based Representation

Dresden Database Research Group

Time series features

- Express global time series behavior
- Reduction to scalar values
- Low computational complexity

Classes of features



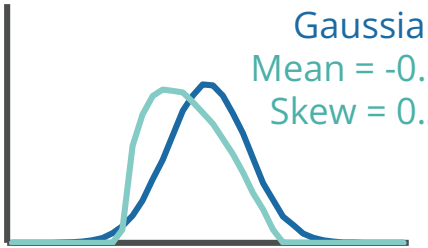
- Distribution
- Correlation
- Components
- Stationarity
- Frequency Domain
- Tests

Kegel, L., Hahmann, M., & Lehner, W. (2018). Feature-based comparison and generation of time series. In *ACM International Conference Proceeding Series*.

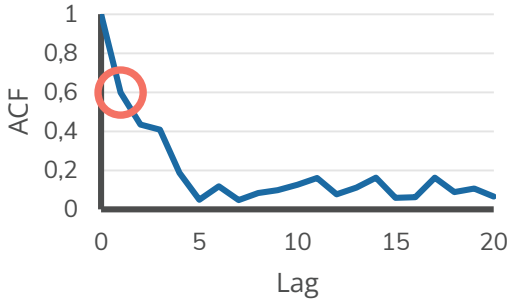
Classes of Features

Dresden Database Research Group

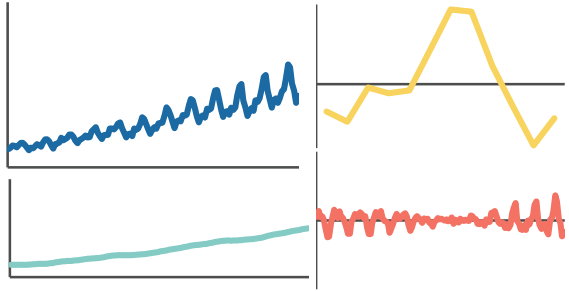
Distribution



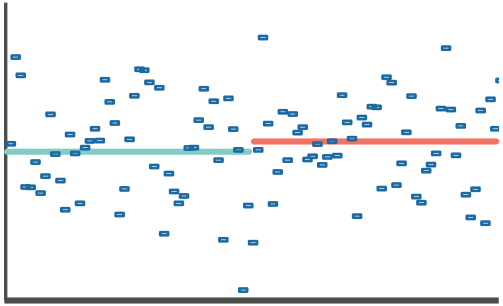
Correlation



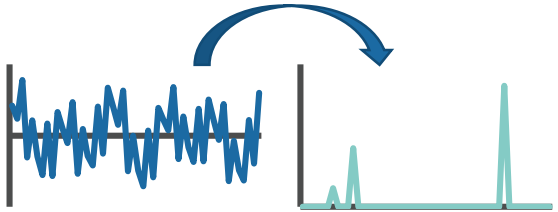
Components



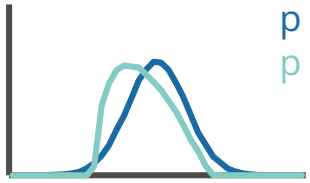
Stationarity



Frequency Domain



Tests



p-value of Normality test

Feature-based Representation

Dresden Database Research Group

Time series features

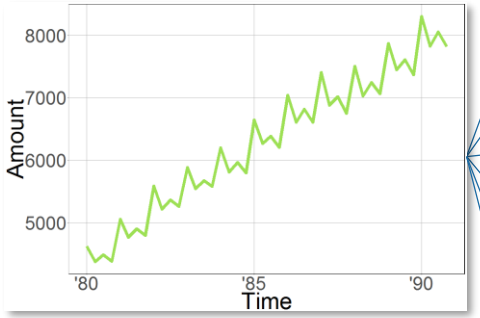
- Express global time series behavior
- Reduction to scalar values
- Low computational complexity

Classes of features

- Distribution: mean, standard deviation, skewness, kurtosis, median, quantiles
- Correlation: autocorrelation coefficients
- Components: trend, season mask, strength
- Stationarity: mean constant across series
- Frequency domain: peaks in periodogram
- Tests: normality test

Advantages

- Dimensionality reduction
- Comparison of time series with different length



Distribution

Correlation

Components

Stationarity

Frequency Domain

Tests

Kegel, L., Hahmann, M., & Lehner, W. (2018). Feature-based comparison and generation of time series. In *ACM International Conference Proceeding Series*.

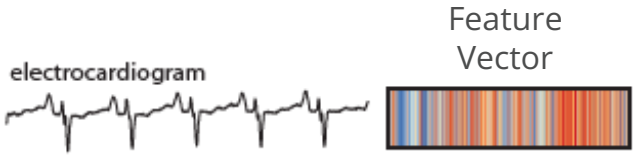
Applications

Time Series Classification

Dresden Database Research Group

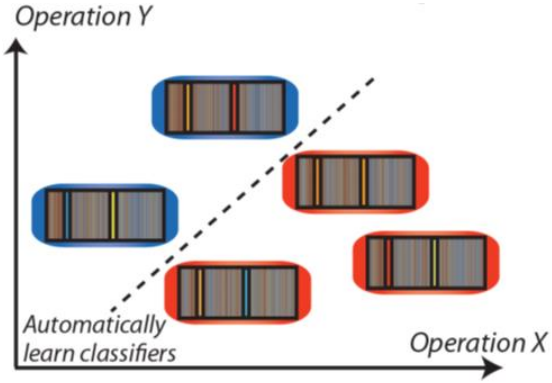
Represent time series by their features

- Features cover large time series domains
- Time Series of different lengths and domains may be reduced to the same representation as a vector of features
- Fulcher reports ~9000 features from the literature



Classification

- Select interesting features with high classification performance
- Build classifiers for further time series or for understanding given time series



Advantages

- Gain insights into differences between classes of labeled time series datasets

Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods..

Time Series Classification (2)

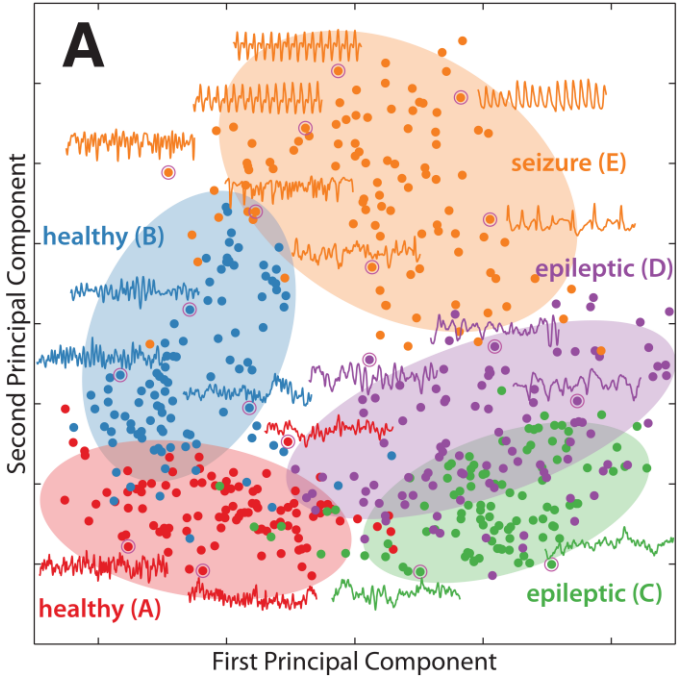
Dresden Database Research Group

Case Study: EEG Recordings

- EEG data (time series) from healthy patients (A, B), epileptic patients (C, D), and patients with epileptic seizure (E)
- Build classifier that differentiates between groups A and E

Results

- Features are reduced to 2 with Principal Component Analysis
- Groups A and E can be well discriminated
- Feature vectors are as performant as other analytical, but more complex time series transformations
  - Support vector machine
  - Discrete wavelet transform



Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods..

Time Series Clustering

Dresden Database Research Group

Goal

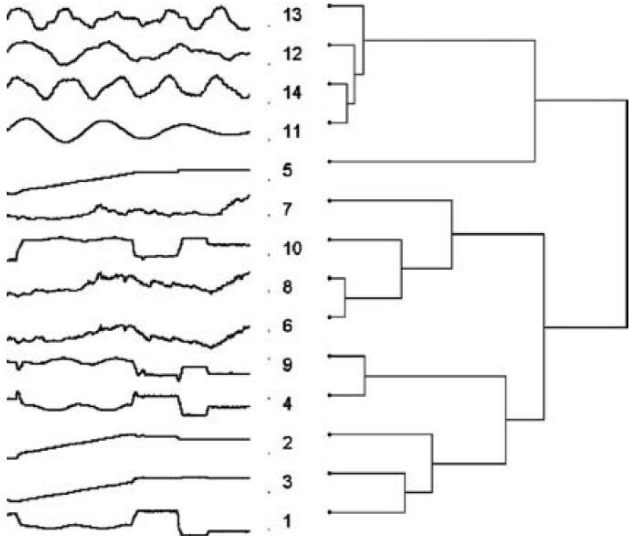
- Partition a set of time series into groups in such a way that homogenous time series are grouped together

Challenges with time series

- Time series data is often to large to fit in main memory
- Time series are highly dimensional and it is difficult for clustering algorithms to handle them
- Their homogeneity strongly depends on the selected similarity measure

Applications

- Which patterns appear frequently?
- Which patterns appear surprisingly?



Wang, X., Smith, K., & Hyndman, R. J. (2006). Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*.

Time Series Clustering (2)

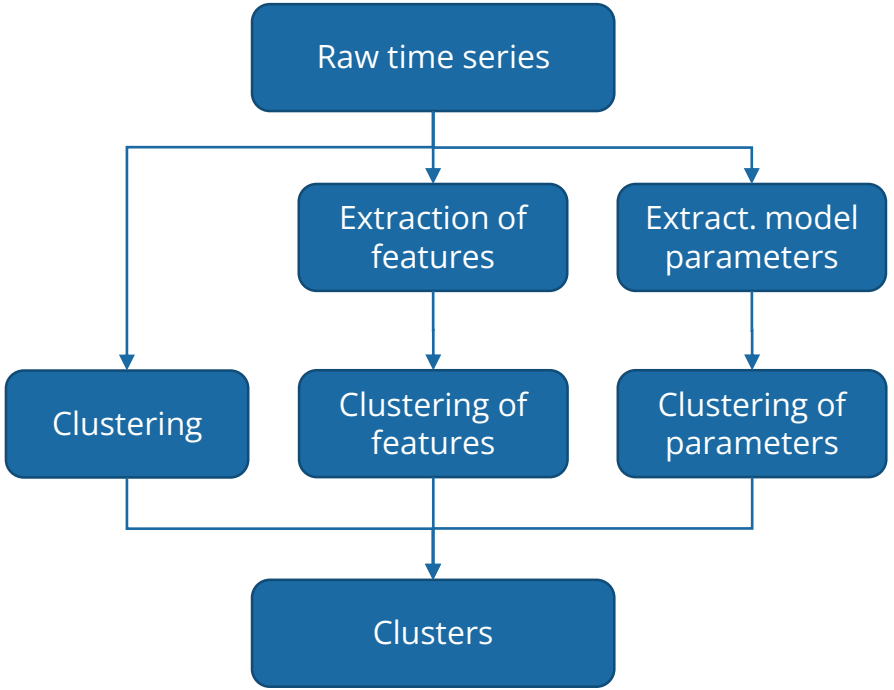
Dresden Database Research Group

Components of time series clustering

- Find appropriate time series representation
- Find appropriate distance measure
- The following components are similar to the usual clustering process

Approaches

- Raw-value-based approach
  - Raw time series is matched as well as possible
  - Usage of stretching and contraction
- Feature-based approach
  - Reduce time series to feature vector
  - Similarity of vector, usually by Euclidean distance
- Model-based approach
  - Fit a model to each given time series
  - Time series distances are calculated based on model parameters



Time Series Forecasting

Classical Time Series Forecasting

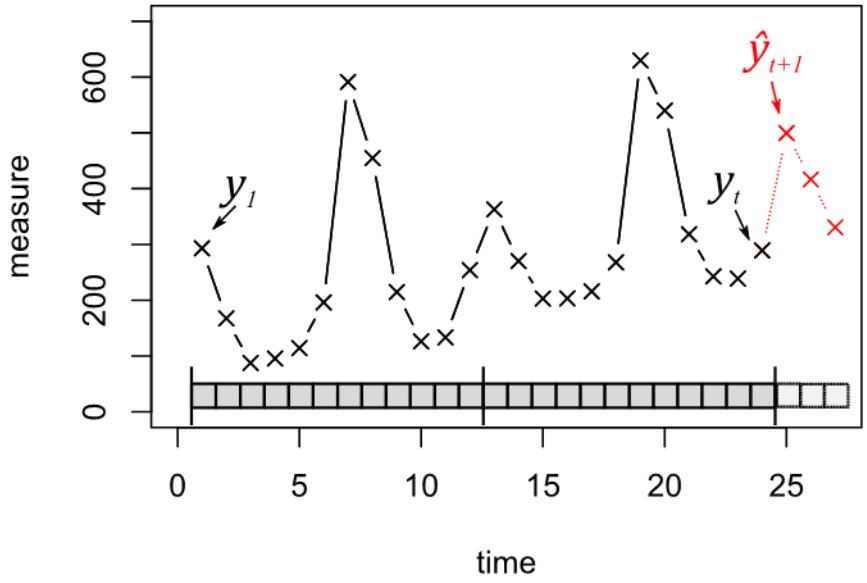
Dresden Database Research Group

Time Series Properties

- Sequence of measure values
- Equidistant
- Complete

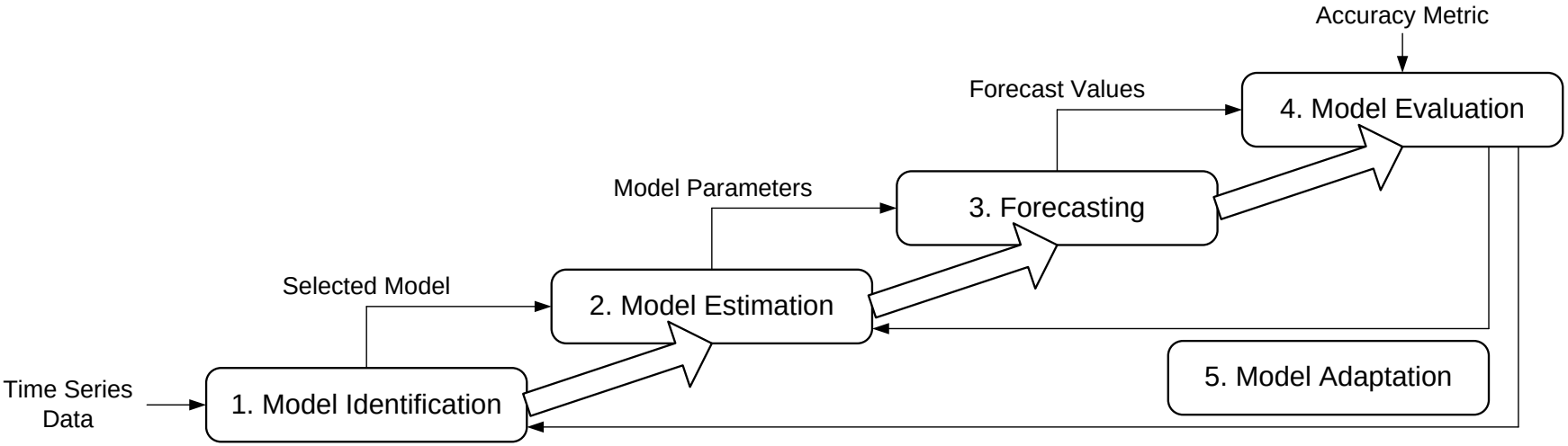
Forecast Models

- One model represents one time series
  - Trend (long-term changes)
  - Season (regular reoccurring changes)
- Accuracy is the only requirement



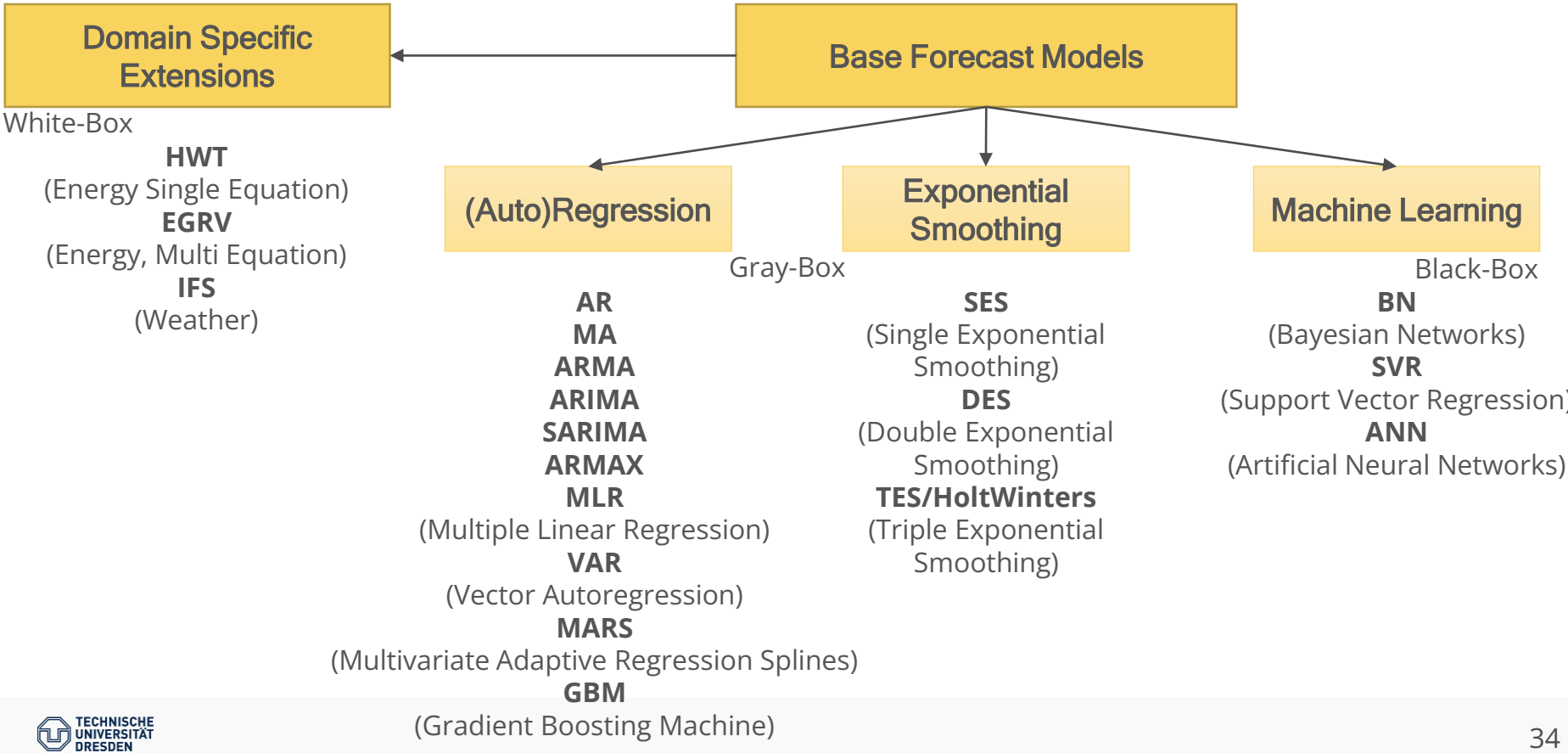
Chatfield, C. (2000). *Time-series forecasting. New York*. Chapman & Hall/CRC Press.

The Forecasting Process



1. **Model Identification** – Choose the optimal model type
2. **Model Estimation** – Instantiation of the model by training it's model parameters
3. **Forecasting** – Usage of the model to calculate the next time series values
4. **Model Evaluation** – Compare the model's predictions with real time series values using an error measure
5. **Model Adaption** – Adaption of the model parameters or the model type

Model Types


Domain Specific Extensions
Base Forecast Models
White-Box
HWT
(Energy Single Equation)
EGRV
(Energy, Multi Equation)
IFS
(Weather)
(Auto)Regression
Exponential Smoothing
Machine Learning
Gray-Box
AR
MA
ARMA
ARIMA
SARIMA
ARMAX
MLR
(Multiple Linear Regression)
VAR
(Vector Autoregression)
MARS
(Multivariate Adaptive Regression Splines)
GBM
(Gradient Boosting Machine)
SES
(Single Exponential Smoothing)
DES
(Double Exponential Smoothing)
TES/HoltWinters
(Triple Exponential Smoothing)
Black-Box
BN
(Bayesian Networks)
SVR
(Support Vector Regression)
ANN
(Artificial Neural Networks)

Exponential Smoothing

Dresden Database Research Group

Weight time series values with smoothing factor

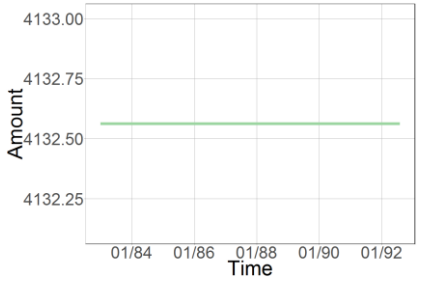
- Declines exponentially for older time series values

Model variants

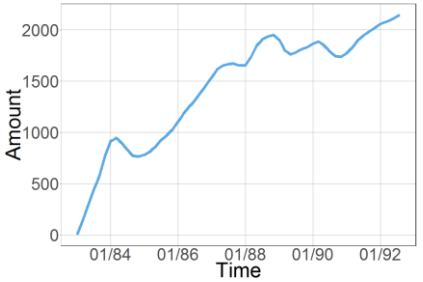
- Simple Exponential Smoothing (SESM): Base
- Double Exponential Smoothing (DESM): Base, Trend
- Triple Exponential Smoothing (TESM): Base, Trend, Season Holt-Winters-Model

Example Base:

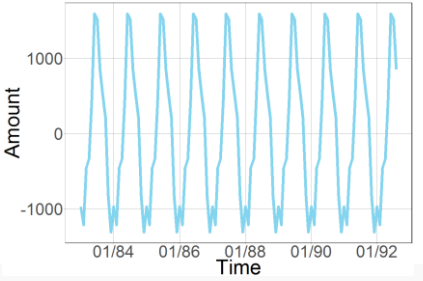
$$x_1^* = x_1$$
$$x_t^* = \alpha \cdot x_t + (1-\alpha) \cdot x_{t-1}^*$$
$$\hat{x}_{t+1} = x_t^*$$



Base



Trend



Season

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1).

Exponential Smoothing

Dresden Database Research Group

Example Single Exponential Smoothing

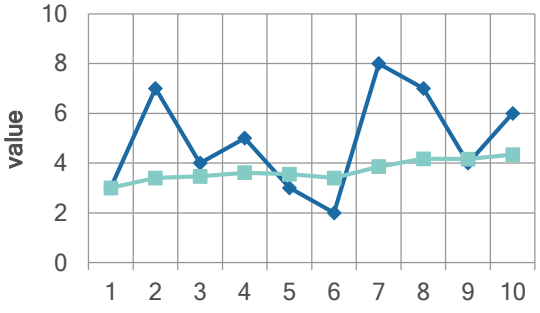
Base: $x_1^* = x_1$

Smoothing: $x_t^* = \alpha \cdot x_t + (1-\alpha) \cdot x_{t-1}^*$

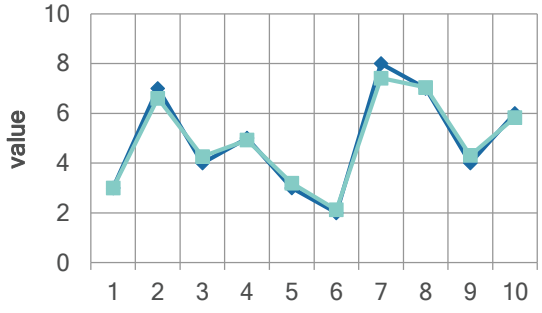
Forecasting: $\hat{x}_{t+1} = x_t^*$

t	1	2	3	4	5	6	7	8	9	10
x_t	3	7	4	5	3	2	8	7	4	6

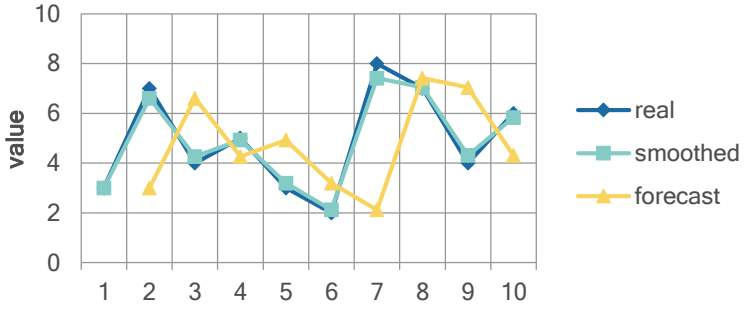
Smoothing
Alpha=0.1



Smoothing
Alpha=0.9



Forecasting
Alpha=0.9



ARMA Models

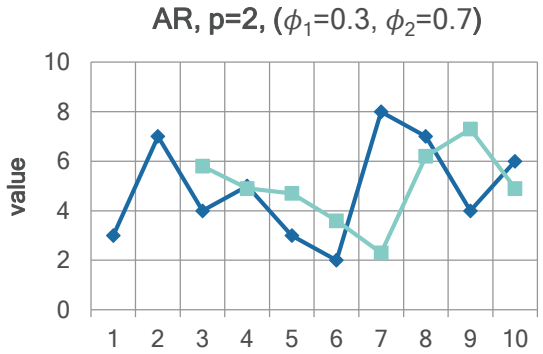
Dresden Database Research Group

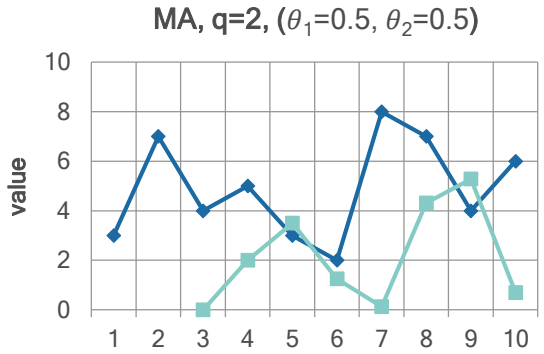
Box and Jenkins Methodology

- Modeling time series with AutoRegression

Classification and Model Hierarchy

- AR(p): $\hat{x}_t = c + \sum_{i=1}^{p} \phi_i \cdot x_{t-i}$
  AutoRegression
- MA(q): $\hat{x}_t = \sum_{i=1}^{q} \theta_i \cdot \varepsilon_{t-i}$
  Moving Average
- $\varepsilon_0, \cdots, \varepsilon_q = 0, \varepsilon_i = x_{t-i} - \hat{x}_{t-i}$





- Combine AR and MA by addition: $\hat{x}_t = \sum_{i=1}^{p} \phi_i \cdot x_{t-i} + \sum_{i=1}^{q} \theta_i \cdot \varepsilon_{t-i} + c$

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (2008). *Time Series Analysis: Forecasting and Control (Wiley Series in Probability and Statistics)*. John Wiley & Sons Inc.

ARIMA Models

Box and Jenkins Methodology

- Modeling time series with AutoRegression

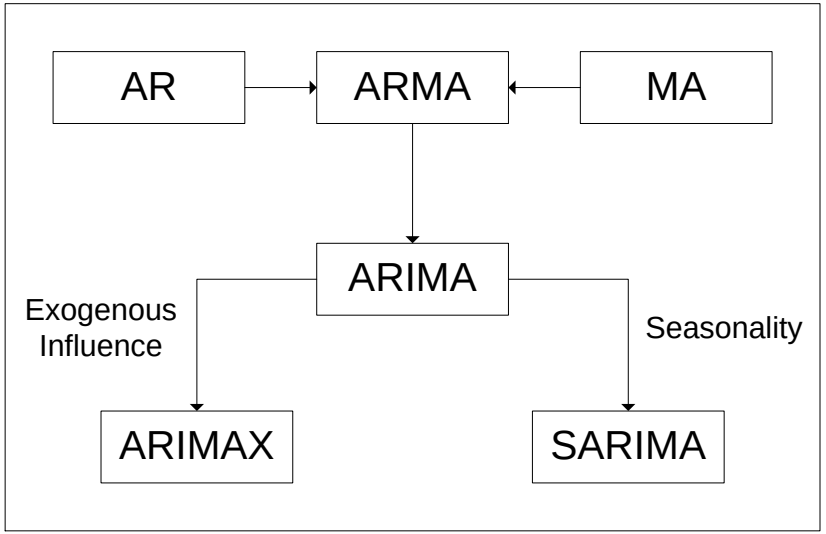
Classification and Model Hierarchy

- AR(p): AutoRegression $\hat{x}_t = c + \sum_{i=1}^{p} \phi_i \cdot x_{t-i}$
- MA(q): $\hat{x}_t = \sum_{i=1}^{q} \theta_i \cdot \varepsilon_{t-i}$

  Moving Average
- ARMA(p,q): combination of AR and MA
- ARIMA(p,d,q): combination von AR, MA, Trend
- AR(I)MAX(p,q,b): ARIMA with exogenous influences

$$\hat{x}_t = \sum_{i=1}^{p} \phi_i \cdot x_{t-i} + \sum_{i=1}^{q} \theta_i \cdot \varepsilon_{t-i} + c + \sum_{i=1}^{b} \gamma_i \cdot W_i$$

- SARIMA(p,d,q)x(P,D,Q)$_s$: ARIMA with seasonality



Vector Autoregression (VAR)

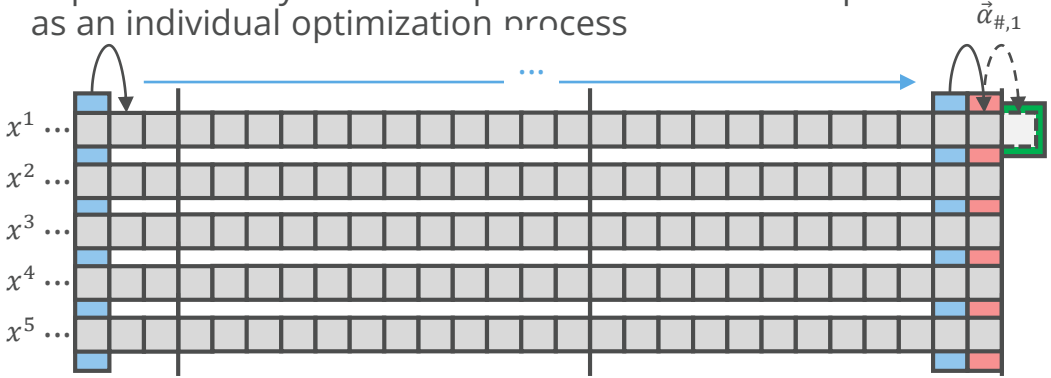
Dresden Database Research Group

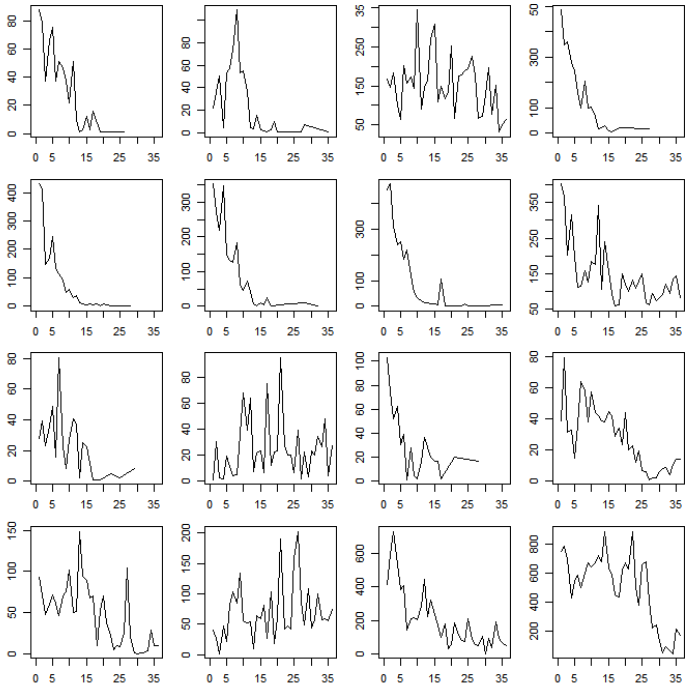
Multivariate Modeling Technique

- Based on the assumption, that time series from the same domain influence each other
- Capable of compensating unexplainable behavior of individual time series
- Can be extended by external influences

Individual Model Optimization

- In practice every line in the parameter matrix is implemented as an individual optimization process





$$\begin{pmatrix} \hat{x}_{1,t+1} \\ \vdots \\ \hat{x}_{n,t+1} \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{n,1} \\ \vdots & \ddots & \vdots \\ \alpha_{1,n} & \cdots & \alpha_{n,n} \end{pmatrix} \cdot \begin{pmatrix} x_{1,t} \\ \vdots \\ x_{n,t} \end{pmatrix}$$

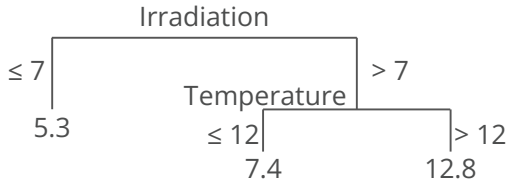
Riise, T., & Tjostheim, D. (1984). Theory and practice of multivariate arma forecasting. *Journal of Forecasting*, 3(3).

Gradient Boosting Machine

Dresden Database Research Group

Boosting

- Combine a set of weak predictors (typically regression trees) into a strong predictor

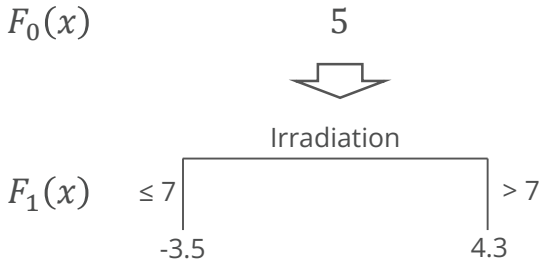


Regression Trees

- Partition data into several groups; represented by the leaves of the tree
- Adjust interaction of variables by number of leaves

Gradient Boosting Machine

- Start with a constant function $F_0(x)$
- Continue to functions F_m (regression trees) to improve the residuals of previous models F_{m-1}
- Stop when the maximum number of iterations is exceeded



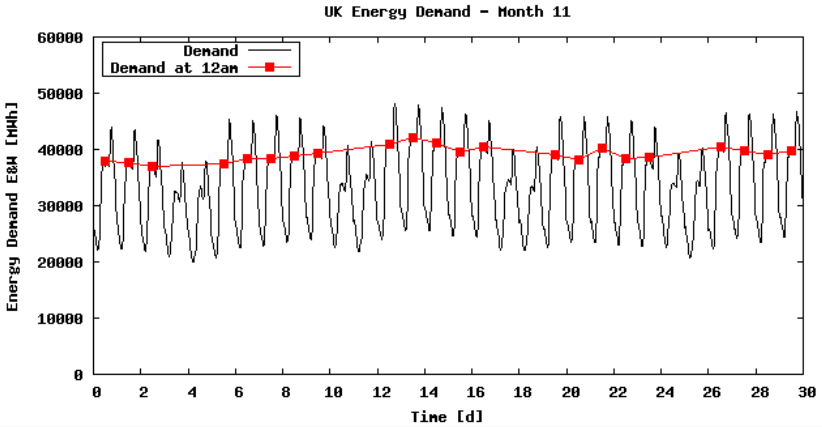
Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5).

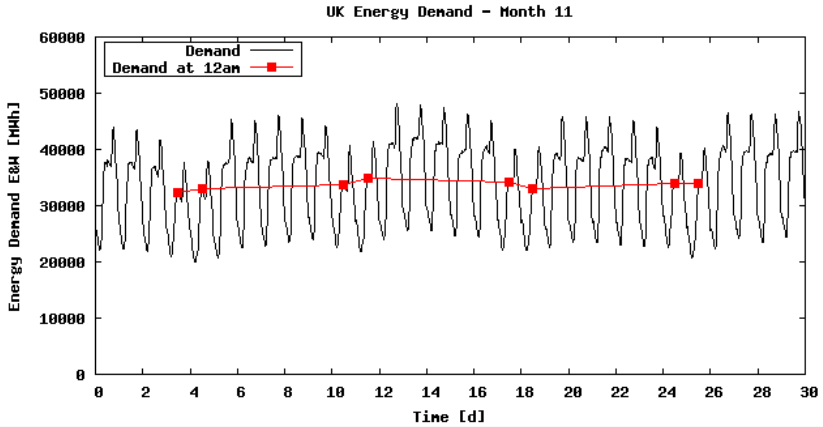
EGRV Model (Engle, Granger, Ramanathan, and Vahid-Arraghi)

Dresden Database Research Group

Multi Equation Model

- One model for each hour (half hour, etc.) of a day (separation in weekdays/weekends)
- White-Box model tailor-made for energy demand
- Decomposition leads to almost constant time series that are easy to predict
- Basic idea works for all types of time series that follow certain pattern





Ramanathan, R., Engle, R., Granger, C. W. J., Vahid-Araghi, F., & Brace, C. (1997). Short-run forecasts of electricity loads and peaks. *International Journal of Forecasting*, *13*(2).

Model Identification

Dresden Database Research Group

(Semi)Automatic Model Identification

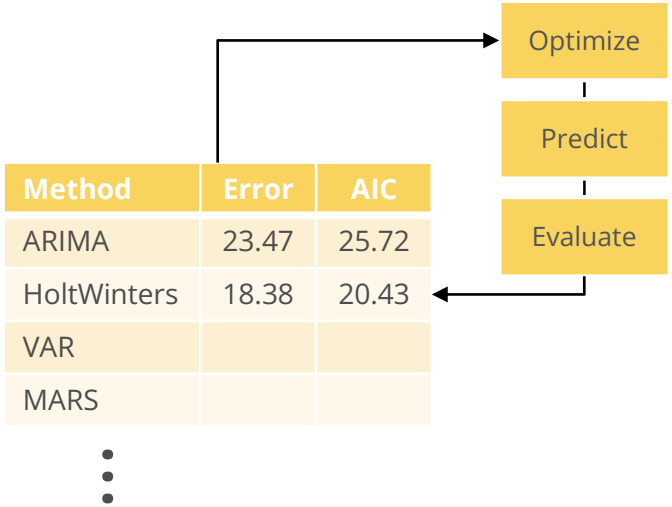
- Test for every available model, how it performs on a test section of the current time series
- Use the error or more complex measures like AIC and BIC to find the optimal model

Akaike Information Criterion (AIC)

- A more sophisticated way to decide which is the optimal model
- Use the fit of the model to the training data and the model complexity
- If two models have an equally good fit, prefer the less complex model

$$AIC = 2k - 2\ln(L)$$

$k = \#\text{parameters}, \ln(L) \approx -\sum_{t=k}^{T} \epsilon_t^2$



Method	Error	AIC
ARIMA	23.47	25.72
HoltWinters	18.38	20.43
VAR		
MARS		

Model Estimation

Dresden Database Research Group

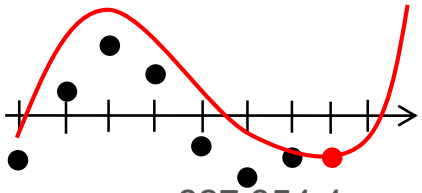
Problem

- Find the model parameters that minimize the error on the training data

Example

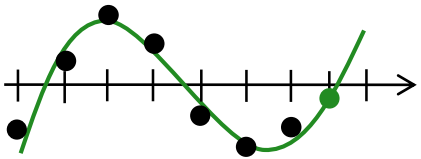
- Forecast Model Type **AR(2)**:
  - $\hat{x}_t = \phi_1 \cdot x_{t-1} + \phi_2 \cdot x_{t-2}$
- Error Metric: **MSE**
  - $\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$
- Parameter Estimator
  - **L-BFGS-B**

Start of Optimization

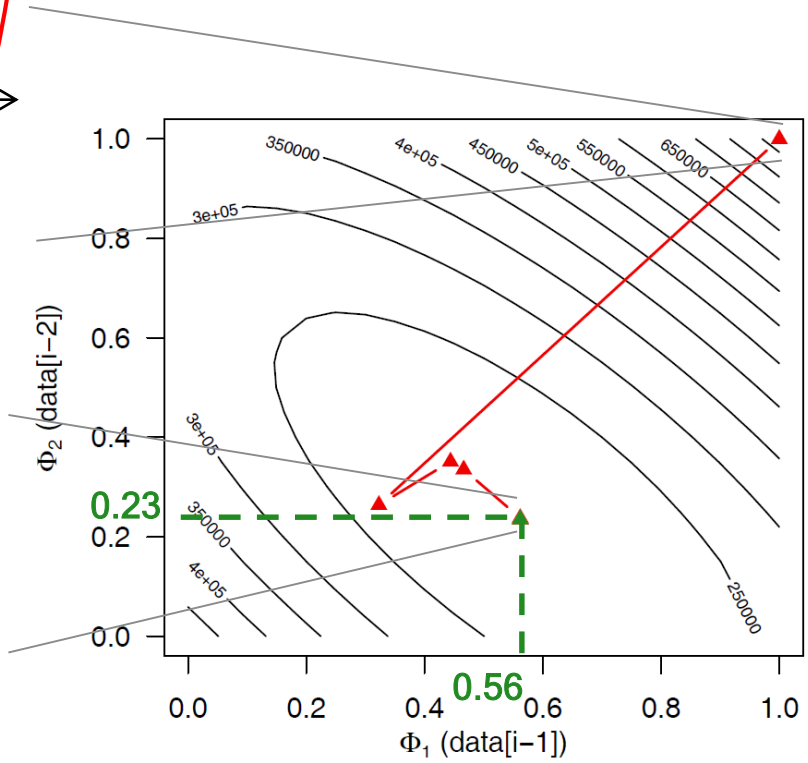


e_{MSE}=827,354.4

End of Optimization



e_{MSE}=211,204.7



Model Usage

Dresden Database Research Group

Model based Prediction

- Application of the identified and trained (optimized) model
- Typically very fast compared to the training process

Horizon

- Describes the number of forecasted values
- One step ahead (only one value), long range forecasting (many values, up to several seasons)

Example (Long Range)

- Single Exponential Smoothing, horizon 10 values
- Continue forecast calculation based on already calculated forecast values

for(i in 1:10)

$$x^*_{t+i-1} = \alpha \cdot \hat{x}_{t+i-1} + (1-\alpha) \cdot x^*_{t-1}$$
$$\hat{x}_{t+i} = x^*_{t+i-1}$$

Base:

$$x^*_1 = x_1$$
$$x^*_t = \alpha \cdot x_t + (1-\alpha) \cdot x^*_{t-1}$$
$$\hat{x}_{t+1} = x^*_t$$

Maintenance Strategies

Dresden Database Research Group

Error Threshold

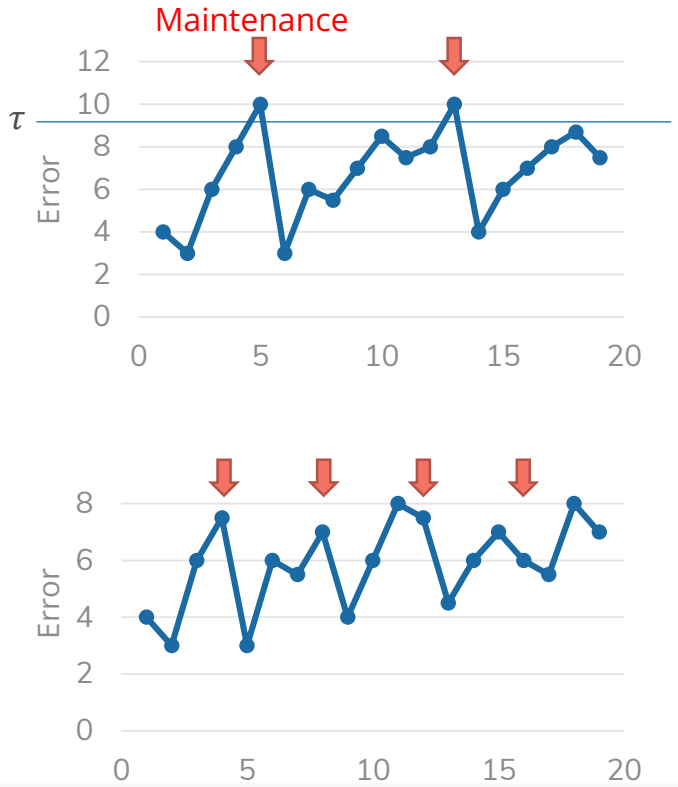
- Define error threshold τ
- Update model parameter when $\text{Error} > \tau$
- May miss opportunities to improve the model performance or reestimate too often, based on whether τ is too high or too low

Time Threshold

- Reestimate the model parameter after a fixed number of update values
- May reestimate too often and without model improvement or tolerate too high errors within an interval

...as usual a combination works best

- A somewhat higher error threshold
- And a somewhat longer time threshold



Time Series Storage

Relational Storage

Dresden Database Research Group

Store Time series values in a relation

- One tuple for each individual measure value
- Store alongside with
  - time/date stamp
  - Time series identifier, e.g.
    - ID
    - Equipment
    - Sensor name

Access

- Usually no order guaranties
- Order by
  - Identifier first
  - Time last

```
select *
from ts
order by ID, rec_date;
```

ID	rec_date	sales
1	20.11.2021	3
1	21.11.2021	7
1	22.11.2021	4
1	23.11.2021	5
1	24.11.2021	3
1	25.11.2021	2
1	26.11.2021	8
1	27.11.2021	7
1	28.11.2021	4
1	29.11.2021	6
2	20.11.2021	7
2	21.11.2021	5
2	22.11.2021	8
⋮		

Relational Storage

Calculating the first derivative

- Discrete differentiation $x'_t = x_t - x_{t-1}$
- Execution in data base:
  - Self join
  - Code time shift in join condition

```
select S.ID, S.rec_date, S.sales-R.sales
from ts as R, ts as S
where R.id=S.id and
      R.rec_date+'1 day'::interval=S.rec_date;
```

ID	rec_date	sales
1	21.11.2021	4
1	22.11.2021	-3
1	23.11.2021	1
1	24.11.2021	-2
1	25.11.2021	-1
1	26.11.2021	6
1	27.11.2021	-1
1	28.11.2021	-3
1	29.11.2021	2
2	21.11.2021	-2
2	22.11.2021	3
2	23.11.2021	1
⋮		

ID	rec_date	sales
1	20.11.2021	3
1	21.11.2021	7
1	22.11.2021	4
1	23.11.2021	5
1	24.11.2021	3
1	25.11.2021	2
1	26.11.2021	8
1	27.11.2021	7
1	28.11.2021	4
1	29.11.2021	6
2	20.11.2021	7
2	21.11.2021	5
2	22.11.2021	8
⋮		

Relational Storage

Dresden Database Research Group

Calculating the first derivative

- Discrete differentiation $x'_t = x_t - x_{t-1}$
- Execution in data base:
  - Self join
  - Code time shift in join condition

Seasonal Differencing

- Shift by one season instead of one value
- $x'_t = x_t - x_{t-s}$

ID	rec_date	sales
1	27.11.2021	4
1	28.11.2021	-3
1	29.11.2021	2
2	27.11.2021	1
2	28.11.2021	4
2	29.11.2021	-2
⋮		

ID	rec_date	sales
1	20.11.2021	3
1	21.11.2021	7
1	22.11.2021	4
1	23.11.2021	5
1	24.11.2021	3
1	25.11.2021	2
1	26.11.2021	8
1	27.11.2021	7
1	28.11.2021	4
1	29.11.2021	6
2	20.11.2021	7
2	21.11.2021	5
2	22.11.2021	8
⋮		

```
select S.ID, S.rec_date, S.sales-R.sales
from ts as R, ts as S
where R.id=S.id and
      R.rec_date+'7 day'::interval=S.rec_date;
```

Relational Storage

Dresden Database Research Group

Autoregressive training tuple generation

- Regression models
  - $\hat{x}_t = f(x_{t-1}, x_{t-2}, \ldots)$
- Example: assume an AR(2) model

```
select S.ID, S.rec_date,
       S.sales as x, R.sales as r1, T.sales as r2
from ts as R, ts as S, ts as T
where T.ID=R.ID and T.ID=S.ID
  and T.rec_date=R.rec_date-'1 day'::interval
  and T.rec_date=S.rec_date-'2 day'::interval
```

ID	rec_date	x	x_{t-1}	x_{t-2}
1	22.11.2021	4	7	3
1	23.11.2021	5	4	7
1	24.11.2021	3	5	4
1	25.11.2021	2	3	5
1	26.11.2021	8	2	3
1	27.11.2021	7	8	2
1	28.11.2021	4	7	8
1	29.11.2021	6	4	7
2	22.11.2021	8	5	7
2	23.11.2021	9	8	5
2	24.11.2021	1	9	8

⋮

ID	rec_date	sales
1	20.11.2021	3
1	21.11.2021	7
1	22.11.2021	4
1	23.11.2021	5
1	24.11.2021	3
1	25.11.2021	2
1	26.11.2021	8
1	27.11.2021	7
1	28.11.2021	4
1	29.11.2021	6
2	20.11.2021	7
2	21.11.2021	5
2	22.11.2021	8

⋮

Array Storage

Dresden Database Research Group

Store entire time series

- Time series values are stored in an array
- Alongside with
  - Time series identifier
  - Start
  - Time interval between values

ID	start	interv	sales
1	20.11.2021	1 day	{3,7,4,5,3,2,8,7,4,6}
2	20.11.2021	1 day	{7,5,8,9,1,5,5,8,9,6}

Constraints

- Time series have to be
  - Complete
  - Equidistant

A Comparison

Dresden Database Research Group

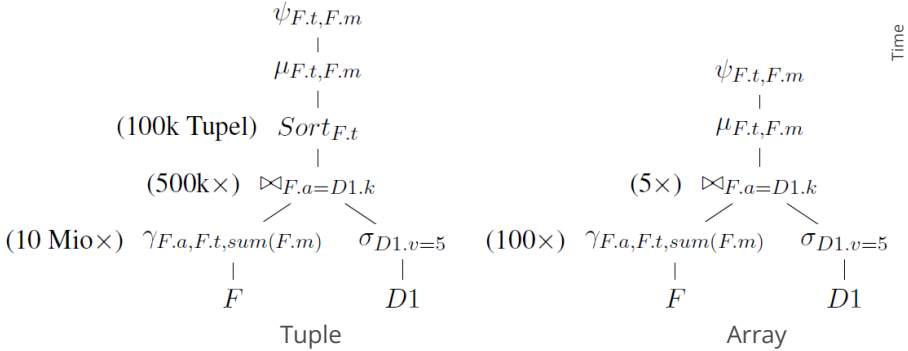
Read performance

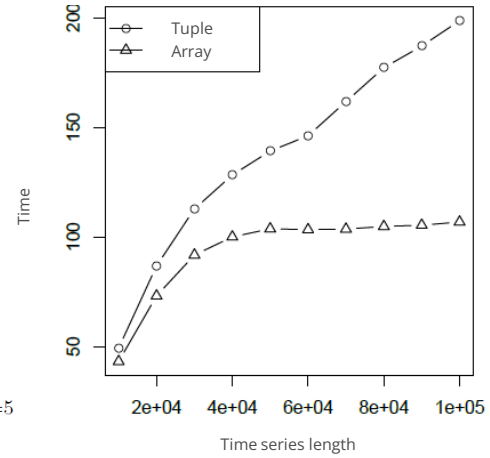
- Read entire data set into a forecast model
- Array storage outperforms tuple per value

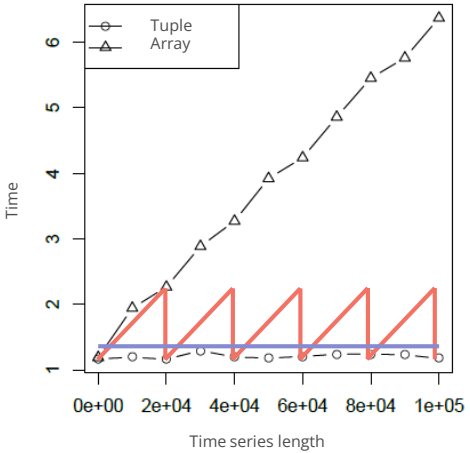
Insert performance

- Append values to stored time series
- Tuple per value wins, less overhead
  - Counter steer by partitioning
- Ideal case, constant insert time

ID	start	interv	sales
1	20.11.2021	1 day	{3,7,4,5,3,2,8,7,4,6}
2	20.11.2021	1 day	{7,5,8,9,1,5,5,8,9,6}







Agenda

Dresden Database Research Group

Basics of Time Series

- Time Series Models
- Decomposition
- Missing Data & Imputation
- Time Series Engineering

Applications

- Time Series Classification
- Time Series Clustering
- Time Series Forecasting

Time Series Storage

- Relational Storage
- Array Storage